Perceiving the World: Question-guided Reinforcement Learning for Text-based Games

Yunqiu Xu[1], **Meng Fang**[2]*, **Ling Chen**[1], **Yali Du**[3], **Joey Tianyi Zhou**[4], **Chengqi Zhang**[1]
[1]University of Technology Sydney, Sydney, Australia
[2]Eindhoven University of Technology, Eindhoven, the Netherlands
[3]King's College London, London, United Kingdom
[4] A*STAR Centre for Frontier AI Research (CFAR), Singapore
{Yunqiu.Xu,Ling.Chen,Chengqi.Zhang}@uts.edu.au, m.fang@tue.nl
yali.du@kcl.ac.uk, zhouty@ihpc.a-star.edu.sg

Abstract

Text-based games provide an interactive way to study natural language processing. While deep reinforcement learning has shown effectiveness in developing the game playing agent, the low sample efficiency and the large action space remain to be the two major challenges that hinder the DRL from being applied in the real world. In this paper, we address the challenges by introducing world-perceiving modules, which automatically decompose tasks and prune actions by answering questions about the environment. We then propose a two-phase training framework to decouple language learning from reinforcement learning, which further improves the sample efficiency. The experimental results show that the proposed method significantly improves the performance and sample efficiency. Besides, it shows robustness against compound error and limited pre-training data.

1 Introduction

Text-based games are simulated environments where the player observes textual descriptions, and acts using text commands (Hausknecht et al., 2020; Urbanek et al., 2019). These games provide a safe and interactive way to study natural language understanding, commonsense reasoning, and dialogue systems. Besides language processing techniques, Reinforcement Learning has become a quintessential methodology for solving text-based games. Some RL-based game agents have been developed recently and proven to be effective in handling challenges such as language representation learning and partial observability (Narasimhan et al., 2015; Fang et al., 2017; Ammanabrolu and Riedl, 2019).

Despite the effectiveness, there are two major challenges for RL-based agents, preventing them from being deployed in real world applications: the *low sample efficiency*, and the *large action space* (Dulac-Arnold et al., 2021). The low sample efficiency is a crucial limitation of RL which refers to the fact that it typically requires a huge amount of data to train an agent to achieve human-level performance (Tsividis et al., 2017). This is because human beings are usually armed with prior knowledge so that they don't have to learn from scratch (Dubey et al., 2018). In a language-informed RL system, in contrast, the agent is required to conduct both language learning and decision making regimes, where the former can be considered as prior knowledge and is much slower than the later (Hill et al., 2021). The sample efficiency could be improved through pre-training methods, which decouple the language learning from decision making (Su et al., 2017). The selection of pre-training methods thus plays an important role: if the pre-trained modules perform poorly on unseen data during RL training, the incurred compound error will severely affect the decision making process. Another challenge is the large discrete action space: the agent may waste both time and training data if attempting irrelevant or inferior actions (Dulac-Arnold et al., 2015; Zahavy et al., 2018).

In this paper, we aim to address these two challenges for reinforcement learning in solving text-based games. Since it is inefficient to train an agent to solve complicated tasks (games) from scratch, we consider decomposing a task into a sequence of subtasks as inspired by (Andreas et al., 2017). We design an RL agent that is capable of automatic task decomposition and subtask-conditioned action pruning, which brings two branches of benefits. First, the subtasks are easier to solve, as the involved temporal dependencies are usually short-term. Second, by acquiring the skills to solve subtasks, the agent will be able to learn to solve a new task more quickly by reusing the learnt skills (Barreto et al., 2020). The challenge of large action space can also be alleviated, if we can filter out the actions that are irrelevant to the current subtask.

Inspired by the observation that human be-

*Corresponding author

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 538 - 560

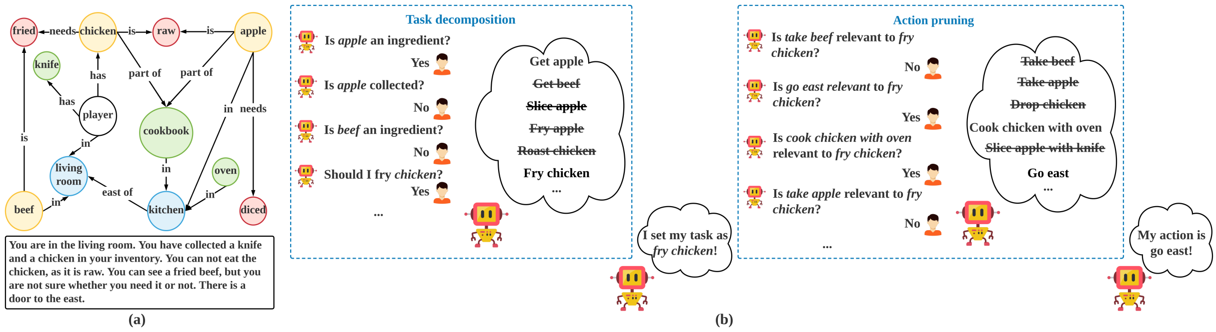

Figure 1: (a) An example of the observation, which can be textual, KG-based, or hybrid. (b) The decision making process. Through question answering, the agent is guided to first decompose the task as subtasks, then reduce the action space conditioned on the subtask.

ings can understand the environment conditions through question answering (Das et al., 2020; Ammanabrolu et al., 2020), we design world-perceiving modules to realize the aforementioned functionalities (i.e., task decomposition and action pruning) and name our method as **Q**uestion-guided **W**orld-perceiving **A**gent (QWA)*. Fig. 1 (b) shows an example of our decision making process. Being guided by some questions, the agent first decomposes the task to obtain a set of available subtasks, and selects one from them. Next, conditioned on the selected subtask, the agent conducts action pruning to obtain a refined set of actions. In order to decouple language learning from decision making, which further improves the sample efficiency, we propose to acquire the world-perceiving modules through supervised pre-training. We design a two-phase framework to train our agent. In the first phase, a dataset is built for the training of the world-perceiving modules. In the second phase, we deploy the agent in games with the pretrained modules frozen, and train the agent through reinforcement learning.

We conduct experiments on a series of cooking games. We divide the games as simple games and complex games, and construct the pre-training dataset from simple games only. The experimental results show that QWA achieves high sample efficiency in solving complex games. We also show that our method enjoys robustness against compound error and limited pre-training data.

Our contributions are summarized as follows: Firstly, we develop an RL agent featured with question-guided task decomposition and action space reduction. Secondly, we design a two-phase framework to efficiently train the agent with limited data. Thirdly, we empirically validate our method's effectiveness and robustness in complex games.

2 Related work

2.1 RL agents for text-based games

The RL agents for text-based games can be divided as text-based agents and KG-based agents based on the form of observations. Compared with the text-based agents (Narasimhan et al., 2015; Yuan et al., 2018; Adolphs and Hofmann, 2020; Jain et al., 2020; Yin and May, 2019; Xu et al., 2020a; Guo et al., 2020), which take the raw textual observations as input to build state representations, the KG-based agents construct the knowledge graph and leverage it as the additional input (Ammanabrolu and Riedl, 2019; Xu et al., 2020b). By providing structural and historical information, the knowledge graph helps the agent to handle partial observability, reduce action space, and improve generalizability across games. Based on how actions are selected, the RL agents can also be divided as parser-based agents, choice-based agents, and template-based agents. The parser-based agents generate actions word by word, leading to a huge combinatorial action space (Kohita et al., 2021). The choice-based agents circumvent this challenge by assuming the access to a set of admissible actions at each game state (He et al., 2016). The template-based agents achieve a trade-off between the huge action space and the assumption of admissible action set by introducing the template-based action space, where the agent selects first a template, and then a verb-object pair either individually (Hausknecht et al., 2020) or conditioned on the selected template (Ammanabrolu and Hausknecht, 2020). In this work, we aim to improve the sam-

*Code is available at: https://github.com/YunqiuXu/QWA ple efficiency and reduce the action space through pre-training. Being agnostic about the form of observations and the action selecting methods, our work complements the existing RL agents.

2.2 Hierarchical RL

Our work is closely related to task decomposition (Oh et al., 2017; Shiarlis et al., 2018; Sohn et al., 2018) and hierarchical reinforcement learning (Dayan and Hinton, 1992; Kulkarni et al., 2016; Vezhnevets et al., 2017). Similar to our efforts, Jiang et al. (2019) and Xu et al. (2021) designed a meta-policy for task decomposition and subtask selection, and a sub-policy for goal-conditioned decision making. Typically, these works either assume the access to a set of available subtasks, or decompose a task through pre-defined rules, while we aim to achieve automatic task decomposition through pre-training, and remove the requirement for expert knowledge during reinforcement learning. Besides, existing work assumes that unlimited interaction data can be obtained to train the whole model through RL. In contrast, we consider the more practical situation where the interaction data is limited, and focus on improving the RL agent's data efficiency. Regarding the sub-policy, we do not assume the access to the termination states of the subtasks. We also do not require additional handcrafted operations in reward shaping (Bahdanau et al., 2019).

2.3 Pre-training methods for RL

There have been a wide range of work studying pre-training methods or incorporating pre-trained modules to facilitate reinforcement learning (Eysenbach et al., 2018; Hansen et al., 2019; Sharma et al., 2019; Gehring et al., 2021; Liu et al., 2021; Schwarzer et al., 2021). One major branch among them is Imitation Learning (IL), where the agent is trained to imitate human demonstrations before being deployed in RL (Hester et al., 2018; Zhu et al., 2018; Reddy et al., 2019). Although we also collect human labeled data for pre-training, we leverage the data to help the agent to perceive the environment instead of learning the solving strategies. Therefore, we do not require the demonstrations to be perfect to solve the game. Besides, our method prevails when pre-trained on simple tasks rather than complicated ones, making it more feasible for human to interact and annotate (Arumugam et al., 2017; Mirchandani et al., 2021). Further discussions to compare our method with IL are provided in subsequent sections.

In the domain of text-based games, some prior works have involved pre-training tasks such as state representation learning (Ammanabrolu et al., 2021; Singh et al., 2021), knowledge graph constructing (Murugesan et al., 2021) and action pruning (Hausknecht et al., 2019; Tao et al., 2018; Yao et al., 2020). For example, Ammanabrolu et al. (2020) designed a module to extract triplets from the textual observation by answering questions, and use these triplets to update the knowledge graph. As far as we know, we are the first to incorporate pre-training based task decompositon in this domain. Besides, instead of directly pruning the actions based on the observation, we introduce subtask-conditioned action pruning to further reduce the action space.

3 Background

POMDP Text-based games can be formulated as a Partially Observable Markov Decision Processes (POMDPs) (Côté et al., 2018). A POMDP can be described by a tuple $\mathcal{G} = \langle \mathcal{S}, \mathcal{A}, P, r, \Omega, O, \gamma \rangle$, with $\mathcal{S}$ representing the state set, $\mathcal{A}$ the action set, $P(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}^+$ the state transition probabilities, $r(s,a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ the reward function, Ω the observation set, O the conditional observation probabilities, and $\gamma \in (0,1]$ the discount factor. At each time step, the agent receives an observation $o_t \in \Omega$ based on the probability $O(o_t|s_t, a_{t-1})$, and select an action $a_t \in \mathcal{A}$. The environment will transit into a new state based on the probability $T(s_{t+1}|s_t, a_t)$, and return a scalar reward r_{t+1}. The goal of the agent is to select the action to maximize the expected cumulative discounted rewards: $R_t = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^k r_t]$.

Observation form In text-based games, the observation can be in the form of text, knowledge graph, or hybrid. Fig. 1 (a) shows an example of the textual observation and the corresponding KG-based observation. We do not make assumptions about the observation form and our method is compatible with any of those forms.

Problem setting We aim to design an RL-based agent that is able to conduct automatic task decomposition and action pruning in solving text-based games. We consider games sharing similar themes and tasks, but varying in their complexities (Adhikari et al., 2020; Chen et al., 2021). Taking the cooking games (Côté et al., 2018) as an example,

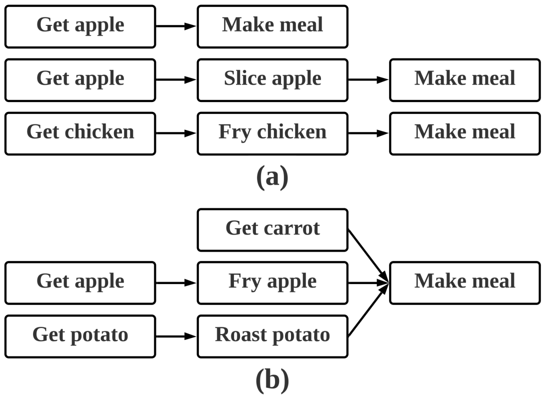

Figure 2: Subtasks for solving (a) 3 simple games and (b) 1 complex game.

the task is always "make the meal". To accomplish this task, the agent has to explore different rooms to collect all ingredients, prepare them in right ways, and make the meal. A game's complexity depends on the number of rooms, ingredients, and the required preparation steps. We define a subtask as a milestone towards completing the task (e.g., "get apple" if "apple" is included in the recipe), and a subtask requires a sequence of actions to accomplish (e.g., the agent has to explore the house to find the apple). A game is considered simple, if it consists of only a few subtasks, and complex if it consists of more subtasks. Fig. 2 gives examples of simple games and complex games. While being closer to real world applications, complex games are hard to solve by RL agents because: 1) it's expensive to collect sufficient human labeled data for pre-training; 2) it's unrealistic to train an RL agent from scratch. We therefore focus on agent's sample efficiency and performance on complex games. Our objective is to leverage the labeled data collected from simple games to speed up RL training in complex games, thus obtaining an agent capable of complex games. For more details and statistics of the simple / complex games used in our work, please refer to Sec. 5.1.

4 Methodology

4.1 Framework overview

Fig. 3 shows the overview of our QWA agent. We consider two world-perceiving modules: a task selector and an action validator. Given the observation o_t and the task candidate set $\mathcal{T}$, we use the task selector to first obtain a subset of currently available subtasks $\mathcal{T}_t \subseteq \mathcal{T}$, then select a subtask $T_t \in \mathcal{T}_t$. Given T_t and the action candidate set $\mathcal{A}$, we use the action validator to get an action subset $\mathcal{A}_t \subseteq \mathcal{A}$, which contains only those relevant to the subtask T_t. Finally, given o_t and T_t, we use an action selector to score each action $a \in \mathcal{A}_t$, and the action with the highest score will be selected as a_t.

The training of the world-perceiving modules can be regarded as the language learning regime, while the training of the action selector can be regarded as the decision making regime. We consider a two-phase training strategy to decouple these two regimes to further improve the sample efficiency (Hill et al., 2021). In the pre-training phase, we collect human interaction data from the simple games, and design QA datasets to train the world-perceiving modules through supervised learning. In the reinforcement learning phase, we freeze the pre-trained modules, and train the action selector in the complex games through reinforcement learning.

4.2 Task selector

Depending on the experiment settings, $\mathcal{T}$ and $\mathcal{A}$ can be either fixed vocabulary sets (parser-based), or changing over time (choice-based). We regard a subtask available if it is essential for solving the "global" task, and there's no prerequisite subtask. For example, the subtask "get apple" in Fig. 1, as the object "apple" is an ingredient which has not been collected. Although another subtask "dice apple" is also essential for making the meal, it is not available since there exists a prerequisite subtask (i.e., you should collect the apple before dicing it). The aim of the task selector is to identify a subset of available subtasks $\mathcal{T}_t \subseteq \mathcal{T}$, and then select one subtask $T_t \in \mathcal{T}_t$.

We formulate the mapping $f(o_t, \mathcal{T}) \rightarrow \mathcal{T}_t$ as a multi-label learning problem (Zhang and Zhou, 2013). For simplicity, we assume that the subtask candidates are independent with each other. Thus, the multi-label learning problem can be decomposed as $|\mathcal{T}|$ binary classification problems. Inspired by the recent progress of question-conditional probing (Das et al., 2020), language grounding (Hill et al., 2021), and QA-based graph construction (Ammanabrolu et al., 2020), we cast these binary classification problems as yes-or-no questions, making the task selector a world-perceiving module. For example, the corresponding question for the subtask candidate "get apple" could be "Whether 'get apple' is an available subtask?". This module can guide the agent to under-

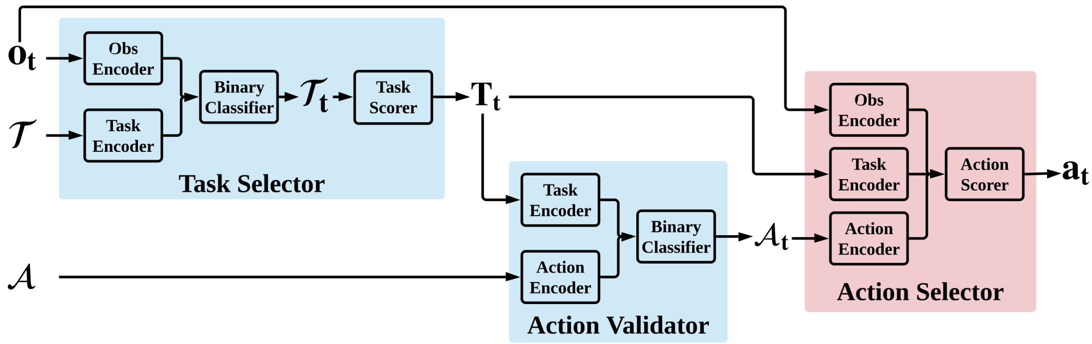

Figure 3: The overview of QWA. The blue modules will be trained in the pre-training phase, while the red module will be trained in the RL phase.

stand the environment conditions through answering questions, but will not directly lead the agent to a specific decision. We can obtain this module through supervised pre-training, and decouple it from reinforcement learning to yield better sample efficiency. Fig. 1 (b) shows some sample QAs, where a human answerer can be replaced by a pre-trained task selector.

Some previous work also considered task decomposition (Chen et al., 2021; Hu et al., 2019), but the related module is obtained through imitating human demonstrations, which is directly related to decision making instead of world perceiving. Compared with these work, our method has two folds of benefits. First, there may exist multiple available subtasks at a timestep. Imitating human demonstrations will specify only one of them, which may be insufficient and lead to information loss. Second, we do not require expert demonstrations which guarantee to solve the game. Instead, we can ask humans to annotate either imperfect demonstrations, or even demonstrations from a random agent. We will treat the IL-based method as a baseline and conduct comparisons in the experiments.

Given the set of available subtasks $\mathcal{T}_t$, arbitrary strategies can be used to select a subtask T_t from it. For example, we can employ a non-learnable task scorer to obtain T_t by random sampling, since each subtask $T \in \mathcal{T}_t$ is essential for accomplishing the task. We can also train a task scorer via a meta-policy for adaptive task selection (Xu et al., 2021).

4.3 Action validator

After obtaining the subtask T_t, we conduct action pruning conditioned on it (or on both T_t and o_t) to reduce the action space, tackling the challenge of large action space. Similar to the task selector, we formulate action pruning as $|\mathcal{A}|$ binary classification problems, and devise another world-perceiving module: the action validator. The action validator is designed to check the relevance of each action candidate $a \in \mathcal{A}$ with respect to T_t by answering questions like "Is the action candidate 'take beef' relevant to the subtask 'fry chicken'?", so as to obtain a subset of actions $\mathcal{A}_t \subseteq \mathcal{A}$ with irrelevant actions filtered. Fig. 3 shows the module architecture. Similar to the task selector, we pre-train this module through question answering. Sample QAs have been shown in Fig. 1 (b).

4.4 Action selector

After pre-training, we deploy the agent in the complex games, and train the action selector through RL. We freeze the pre-trained modules, as no human labeled data will be obtained in this phase. At each time step, we use the task selector and the action validator to produce $\mathcal{T}_t$ and $\mathcal{A}_t$, respectively. We keep using the same subtask T over time until it is not included in $\mathcal{T}_t$, as we do not want the agent to switch subtasks too frequently. The agent can simply treat T_t as the additional observation of o_t. If we do not limit the use of human knowledge in this phase, we can also treat T_t as a goal with either hand-crafted (Jiang et al., 2019) or learnt reward function (Colas et al., 2020). Arbitrary methods can be used for optimizing (Ammanabrolu and Hausknecht, 2020; Adhikari et al., 2020).

One issue we are concerned about is the compound error – the prediction error from imperfect pre-trained modules will adversely affect RL training (Talvitie, 2014; Racanière et al., 2017). For example, the false predictions made by the binary classifier in the task selector may lead to a wrong T_t, which affects $\mathcal{A}_t$ and a_t in turn. To alleviate the influence of the compound error, we assign time-awareness to subtasks. A subtask is bounded

Table 1: Game statistics. We use the simple games to provide human labeled data in the pre-training phase. We use the medium & hard games in the reinforcement learning phase.

Name	Traj.Length	#Triplets	#Rooms	#Objs	#Ings	#Reqs	#Acts	#Subtasks	#Avail.Subtasks
Simple	7.90	38.48	5.76	23.69	1.49	0.96	14.50	12.44	1.14
Medium	15.30	51.07	6.00	26.10	3.00	3.00	23.48	23.00	1.94
Hard	21.75	59.95	8.00	31.48	3.00	4.00	22.94	23.00	2.16

by a time limit $[0, \xi]$. If the current subtask T is not finished within its time limit, we force the agent to re-select a new subtask $T_t \in \mathcal{T}_t \setminus \{T\}$, regardless whether T is still available. Besides making the agent robust against errors, another benefit by introducing time-awareness to subtasks is that it improves the subtask selection diversity, which helps the agent to avoid getting stuck in local minima (Pong et al., 2020; Campero et al., 2020).

5 Experiments

5.1 Experiment settings

We conduct experiments on cooking games provided by the rl.0.2 game set[†] and the FTWP game set[‡], which share the vocabulary set. Based on the number of subtasks, which is highly correlated to the number of ingredients & preparing requirements, we design three game sets with varying complexities: 3488 simple games, 280 medium games and 420 hard games. Note that there is no overlapping games between the simple set and the medium / hard game sets. Table 1 shows the game statistics. Besides "Traj.Length", which denotes the average length of the expert demonstrations per game[§], other statistic metrics are averaged per time step per game (e.g., "#Subtasks" and "#Avail.Subtasks" denote the average number of subtask candidates $\mathcal{T}$, and the average number of available subtasks $\mathcal{T}_t$, respectively). We will collect human interaction data from the simple games for pre-training. We regard both medium & hard games as complex, and will conduct reinforcement learning on these two game sets without labeled data.

5.2 Baselines

We consider the following four models, and compare with more variants in ablation studies:

- **GATA** (Adhikari et al., 2020): a powerful KG-based RL agent, which is the benchmark model for cooking games.
- **IL** (Chen et al., 2021): a hierarchical agent which also uses two training phases. In the first phase, both the task selector and the action selector are pre-trained through imitation learning. Then in the second phase, the action selector is fine-tuned through reinforcement learning.
- **IL w/o FT**: a variant of the IL baseline, where only the imitation pre-training phase is conducted, and there's no RL fine-tuning.
- **QWA**: the proposed model with world-perceiving modules.

5.3 Implementation details

Model architecture All models are implemented based on GATA's released code[¶]. In particular, we use the version GATA-GTF, which takes only the KG-based observation, and denote it as GATA for simplicity. The observation encoder is implemented based on the Relational Graph Convolutional Networks (R-GCNs) (Schlichtkrull et al., 2018) by taking into account both nodes and edges. Both the task encoder and the action encoder are implemented based on a single transformer block with single head (Vaswani et al., 2017) to encode short texts. The binary classifier, the task scorer and the action scorer are linear layers. The GATA and IL models are equipped with similar modules. Please refer to Appendix C for details.

Pre-training We train the task selector and the action validator separately, as they use different types of QAs. We ask human players to play the simple games, and answer the yes-or-no questions based on the observations. The details of the dataset construction (interaction data collection, question generation, answer annotation, etc.) could be found at Appendix B. We train the task selector with a batch size of 256, and the action

[†] https://aka.ms/twkg/rl.0.2.zip

[‡] https://aka.ms/ftwp/dataset.zip

[§] The demonstrations of the medium & hard games are just for statistics, and will not be used for pre-training.

[¶] https://github.com/xingdi-eric-yuan/GATA-public

Table 2: The testing performance at 20% / 100% of the reinforcement learning phase.

Model	Medium		Hard	
	20%	100%	20%	100%
QWA (ours)	**0.66**±0.02	**0.71**±0.04	**0.53**±0.04	**0.53**±0.02
GATA	0.31±0.02	0.57±0.18	0.25±0.02	0.48±0.01
IL	0.45±0.18	0.26±0.03	0.32±0.11	0.35±0.08
IL w/o FT	0.63±0.05	0.63±0.05	0.48±0.05	0.48±0.05

validator with a batch size of 64. The modules are trained for 10-20 epochs using Focal loss and Adam optimizer with a learning rate of 0.001.

Reinforcement learning We consider the medium game set and hard game set as different experiments. We split the medium game set into 200 training games / 40 validation games / 40 testing games, and the hard game set into 300 / 60 / 60. We follow the default setting of (Adhikari et al., 2020) to conduct reinforcement learning. We set the step limit of an episode as 50 for training and 100 for validation / testing. We set the subtask time limit $\xi = 5$. For each episode, we sample a game from the training set to interact with. We train the models for 100,000 episodes. The models are optimized via Double DQN (epsilon decays from 1.0 to 0.1 in 20,000 episodes, Adam optimizer with a learning rate of 0.001) with Pritorized Experience Replay (replay buffer size 500,000). For every 1,000 training episodes, we validate the model and report the testing performance.

5.4 Evaluation metrics

We measure the models through their RL testing performance. We denote a game's score as the episodic sum of rewards without discount. As different games may have different maximum available scores, we report the normalized score, which is defined as the collected score normalized by the maximum score for a game.

6 Results and discussions

6.1 Main results

Fig. 4 shows the RL testing performance with respect to the training episodes. Table 2 shows the testing performance after 20,000 training episodes (20%) / at the end of RL training (100%). Compared with GATA, which needs to be "trained from scratch", the proposed QWA model achieves high sample efficiency: it reaches convergence with high performance before 20% of the training stage,

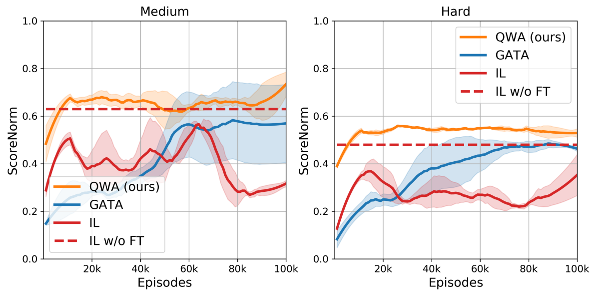

Figure 4: The RL testing performance w.r.t. training episodes. The red dashed line denotes the IL agent without fine-tuning.

saving 80% of the online interaction data in complex games. The effectiveness of pre-training can also be observed from the variant "IL w/o FT": even though it requires no further training on the medium / hard games, it achieves comparable performance to our model. However, the performance of QWA can be further improved through RL, while it does not work for the IL-based model, as we can observe the performance of "IL" becomes unstable and drops significantly during the RL fine-tuning. A possible reason is that there exists large domain gap between simple and medium (hard) games, and our model is more robust against such domain shifts. For example, our world-perceiving task selector performs better than IL-based task selector in handling more complex observations (according to Table 1, the observations in medium / hard games contain more triplets, rooms and objects), facilitating the training of the action selector. Besides the domain gap in terms of the observation space, there is also a gap between domains in terms of the number of available subtasks – while there's always one available subtask per time step in simple games, the model will face more available subtasks in the medium / hard games. Different from our task selector, which is trained to check the availability of every subtask candidate, the IL pre-trained task selector can not adapt well in this situation, as it is trained to find the unique subtask and ignore the other subtask candidates despite whether they are also available.

6.2 Performance on the simple games

We further investigate the generalization performance of our model on simple games, considering that simple games are not engaged in our RL training. To conduct the experiment, after RL training, we deploy all models on a set of 140 held-out sim-

Table 3: The RL testing performance on simple games.

Model	Medium 100%	Hard 100%
QWA (ours)	**0.80**±0.01	**0.82**±0.02
GATA	0.32±0.03	0.45±0.12
IL	0.44±0.02	0.29±0.03
IL w/o FT	0.76±0.06	0.76±0.06

ple games for RL interaction. Table 3 shows the results, where "Medium 100%" ("Hard 100%") denotes that the model is trained on medium (hard) games for the whole RL phase. The generalizability of GATA, which is trained purely with medium and hard games, is significantly low and cannot perform well on simple games. In contrast, our model performs very well and achieves over 80% of the scores. The world-perceiving modules, which are pre-trained with simple games, help to train a decision module that adapts well on unseen games. It is not surprising that the variant "IL w/o FT" also performs well on simple games, since they are only pre-trained with simple games. However, as indicated by the performance of "IL", after fine-tuning on medium/hard games (recalling Sec. 6.1), the action scorer "forgets" the experience/skills dealing with simple games and the model fails to generalize on unseen simple games. In summary, the best performance achieved by QWA demonstrates that our model can generalize well on games with different complexities.

6.3 Ablation study

We study the contribution of the subtask time-awareness by comparing our full model with the variant without this technique. Fig. 5 shows the result. Although the models perform similarly in the medium games, the full model shows better performance in the hard games, where there may exist more difficult subtasks (we regard a subtask more difficult if it requires more actions to be completed). Assigning each subtask a time limit prevents the agent from pursuing a too difficult subtask, and improves subtask diversity by encouraging the agent to try different subtasks. Besides, it prevents the agent from being stuck in a wrong subtask, making the agent more robust to the compound error.

We then investigate the performance upper bound of our method by comparing our model to variants with oracle world-perceiving modules. Fig. 6 shows the results, where "+expTS" ("+expAV") denotes that the model uses an expert task selector (action validator). There's still space to improve the

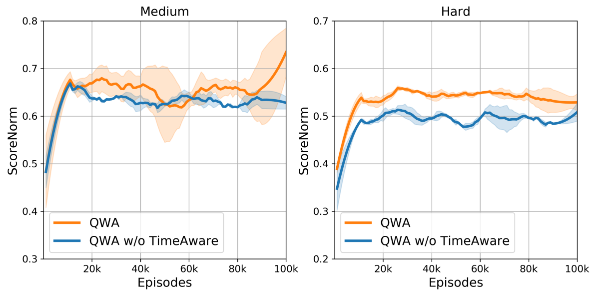

Figure 5: The performance of our model and the variant without time-awareness.

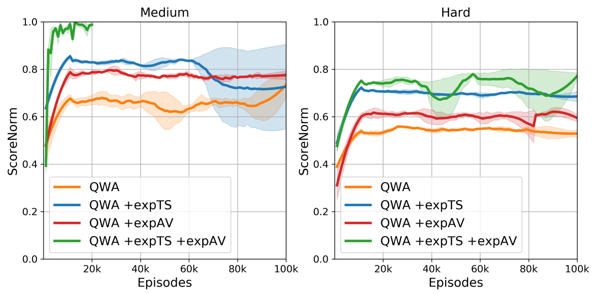

Figure 6: The performance of our model and the variants with expert modules.

pre-trained modules. The variant "QWA +expTS +expAV" solves all the medium games and achieves nearly 80% of the scores in hard games, showing the potential of introducing world-perceiving modules in facilitating RL. We also find that assigning either the expert task selector or the expert action validator helps to improve the performance. In light of these findings, we will consider more powerful pre-training methods as a future direction.

6.4 Pre-training on the partial dataset

Although we only collect labeled data from the simple games, it is still burdensome for human players to go through the games and answer the questions. We are thus interested in investigating how the performance of our QWA (or world-perceiving modules) varies with respect to a reduced amount of pre-training data. Fig. 7 shows the results, where the pre-training dataset has been reduced to 75%, 50% and 25%, respectively. Our model still performs well when the pre-training data is reduced to 75% and 50%. When we only use 25% of the pre-training data, the model exhibits instability during the learning of hard games. Being pre-trained on a largely-reduced dataset, the world-perceiving modules might be more likely to make wrong predictions with the progress of RL training, leading to the performance fluctuation. However, the fi-

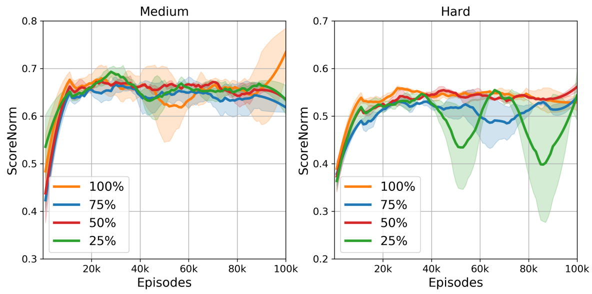

Figure 7: The performance of our model with varying amounts of pre-training data.

nal performance of this variant is still comparable. To summarize, our model is robust to limited pre-training data and largely alleviates the burden of human annotations.

7 Conclusion

In this paper, we addressed the challenges of low sample efficiency and large action space for deep reinforcement learning in solving text-based games. We introduced the world-perceiving modules, which are capable of automatic task decomposition and action pruning through answering questions about the environment. We proposed a two-phase training framework, which decouples the language learning from the reinforcement learning. Experimental results show that our method achieves improved performance with high sample efficiency. Besides, it shows robustness against compound error and limited pre-training data. Regarding the future work, we would like to further improve the pre-training performance by introducing contrastive learning objective (You et al., 2020) and KG-based data augmentation (Zhao et al., 2021).

Acknowledgements

This work is supported in part by ARC DP21010347, ARC DP180100966 and Facebook Research. Joey Tianyi Zhou is supported by A*STAR SERC Central Research Fund (Use-inspired Basic Research). We thank the anonymous reviewers for their constructive suggestions. We thank Smashicons and Trazobanana for providing the icons in Fig. 1.

References

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and Will Hamilton. 2020. Learning dynamic belief graphs to generalize on text-based games. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 3045–3057.

Leonard Adolphs and Thomas Hofmann. 2020. Ledeepchef: Deep reinforcement learning agent for families of text-based games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 7342–7349.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations (ICLR)*.

Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, pages 3557–3565.

Prithviraj Ammanabrolu, Ethan Tien, Zhaochen Luo, and Mark O Riedl. 2020. How to avoid being eaten by a grue: Exploration strategies for text-adventure agents. *arXiv preprint arXiv:2002.08795*.

Prithviraj Ammanabrolu, Jack Urbanek, Margaret Li, Arthur Szlam, Tim Rocktäschel, and Jason Weston. 2021. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 807–833.

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning (ICML)*, volume 70, pages 166–175.

Dilip Arumugam, Siddharth Karamcheti, Nakul Gopalan, Lawson LS Wong, and Stefanie Tellex. 2017. Accurately and efficiently interpreting human-robot instructions of varying granularities. In *Proceedings of Robotics: Science and Systems (RSS)*.

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. 2019. Learning to understand goal specifications by modelling reward. In *International Conference on Learning Representations (ICLR)*.

André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. 2020. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087.

Andres Campero, Roberta Raileanu, Heinrich Kuttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. 2020. Learning with amigo: Adversarially motivated intrinsic goals. In *International Conference on Learning Representations (ICLR)*.

Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. 2019. Learning action representations for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 941–950. PMLR.

Yash Chandak, Georgios Theocharous, Chris Nota, and Philip Thomas. 2020. Lifelong learning with a changing action set. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 3373–3380.

Valerie Chen, Abhinav Gupta, and Kenneth Marino. 2021. Ask your humans: Using human instructions to improve generalization in reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Dominey, and Pierre-Yves Oudeyer. 2020. Language as a cognitive tool to imagine goals in curiosity driven exploration. *Advances in Neural Information Processing Systems (NeurIPS)*, 33.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Abhishek Das, Federico Carnevale, Hamza Merzic, Laura Rimell, Rosalia Schneider, Josh Abramson, Alden Hung, Arun Ahuja, Stephen Clark, Greg Wayne, et al. 2020. Probing emergent semantics in predictive agents via question answering. In *International Conference on Machine Learning (ICML)*, pages 2376–2391. PMLR.

Peter Dayan and Geoffrey E Hinton. 1992. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 5.

Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Tom Griffiths, and Alexei Efros. 2018. Investigating human priors for playing video games. In *International Conference on Machine Learning (ICML)*, pages 1349–1357. PMLR.

Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, pages 1–50.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2018. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 595–605.

Jonas Gehring, Gabriel Synnaeve, Andreas Krause, and Nicolas Usunier. 2021. Hierarchical skills for efficient exploration. *Advances in Neural Information Processing Systems (NeurIPS)*, 34.

Xiaoxiao Guo, Mo Yu, Yupeng Gao, Chuang Gan, Murray Campbell, and Shiyu Chang. 2020. Interactive fiction game playing as multi-paragraph reading comprehension with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7755–7765.

Steven Hansen, Will Dabney, Andre Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. 2019. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations (ICLR)*.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 7903–7910.

Matthew Hausknecht, Ricky Loynd, Greg Yang, Adith Swaminathan, and Jason D Williams. 2019. Nail: A general interactive fiction agent. *arXiv preprint arXiv:1902.04259*.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1621–1630.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. 2018. Deep q-learning from demonstrations. In *Thirty-second AAAI conference on artificial intelligence (AAAI)*.

Felix Hill, Olivier Tieleman, Tamara von Glehn, Nathaniel Wong, Hamza Merzic, and Stephen Clark. 2021. Grounded language learning fast and slow. In *International Conference on Learning Representations (ICLR)*.

Hengyuan Hu, Denis Yarats, Qucheng Gong, Yuandong Tian, and Mike Lewis. 2019. Hierarchical decision making by generating and following natural language instructions. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:10025–10034.

Vishal Jain, William Fedus, Hugo Larochelle, Doina Precup, and Marc G Bellemare. 2020. Algorithmic improvements for deep reinforcement learning applied to interactive fiction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 4328–4336.

Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. 2019. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 9419–9431.

Ryosuke Kohita, Akifumi Wachi, Daiki Kimura, Subhajit Chaudhury, Michiaki Tatsubori, and Asim Munawar. 2021. Language-based general action template for reinforcement learning agents. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2125–2139.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems (NeurIPS)*, 29:3675–3683.

Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, SM Eslami, Daniel Hennes, Wojciech M Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, et al. 2021. From motor control to team play in simulated humanoid football. *arXiv preprint arXiv:2105.12196*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2017. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Suvir Mirchandani, Siddharth Karamcheti, and Dorsa Sadigh. 2021. Ella: Exploration through learned language abstraction. *Advances in Neural Information Processing Systems (NeurIPS)*, 34.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2021. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 9018–9027.

Karthik Narasimhan, Tejas D Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–11.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 70, pages 2661–2670.

Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. 2020. Skew-fit: State-covering self-supervised reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 7783–7792. PMLR.

Sébastien Racanière, Théophane Weber, David P Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. 2017. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5694–5705.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. 2019. Sqil: Imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations (ICLR)*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference (ESWC)*, pages 593–607.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. 2021. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. 2019. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*.

Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. 2018. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning (ICML)*, volume 80, pages 4654–4663.

Ishika Singh, Gargi Singh, and Ashutosh Modi. 2021. Pre-trained language models as prior knowledge for playing text-based games. *arXiv preprint arXiv:2107.08408*.

Sungryull Sohn, Junhyuk Oh, and Honglak Lee. 2018. Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 7156–7166.

Pei-Hao Su, Paweł Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 147–157.

Erik Talvitie. 2014. Model regularization for stable sample rollouts. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 780–789.

Ruo Yu Tao, Marc-Alexandre Côté, Xingdi Yuan, and Layla El Asri. 2018. Towards solving text-based games by producing adaptive action spaces. *arXiv preprint arXiv:1812.00855*.

Pedro A Tsividis, Thomas Pouncy, Jaqueline L Xu, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Human learning in atari. In *2017 AAAI Spring Symposium Series*.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 673–683.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5998–6008.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. FeUdal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 70, pages 3540–3549.

Yunqiu Xu, Ling Chen, Meng Fang, Yang Wang, and Chengqi Zhang. 2020a. Deep reinforcement learning with transformers for text adventure games. In *IEEE Conference on Games (CoG)*, pages 65–72.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, and Chengqi Zhang. 2021. Generalization in text-based games via hierarchical reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1343–1353.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Tianyi Zhou, and Chengqi Zhang. 2020b. Deep reinforcement learning with stacked hierarchical attention for text-based games. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 16495–16507.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754.

Xusen Yin and Jonathan May. 2019. Comprehensible context-driven text game playing. *IEEE Conference on Games (CoG)*, pages 1–8.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:5812–5823.

Xingdi (Eric) Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2018. Counting to explore and generalize in text-based games. In *European Workshop on Reinforcement Learning (EWRL)*.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. 2018. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 3562–3573.

Min-Ling Zhang and Zhi-Hua Zhou. 2013. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021. Data augmentation for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 11015–11023.

Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, and Nicolas Heess. 2018. Reinforcement and imitation learning for diverse visuomotor skills. In *Proceedings of Robotics: Science and Systems (RSS)*.

Appendix

The appendix is organized as follows: Sec. A details the environment. Sec. B illustrates the process for constructing the pre-training datasets. Sec. C demonstrates the baselines' architecture and training details. Sec. D provides more experimental results.

A Game Environment

In the cooking game (Côté et al., 2018), the player is located in a house, which contains multiple rooms and interactable objects (food, tools, etc.). Her / his task is to follow the recipe to prepare the meal. Each game instance has a unique recipe, including different numbers of ingredients (food objects that are necessary for preparing the meal) and their corresponding preparation requirements (e.g., "slice", "fry"). Besides the textual observation, the KG-based observation can also be directly obtained from the environment. The game sets used in our work contains a task set $\mathcal{T}$ of 268 subtasks, and an action set $\mathcal{A}$ of 1304 actions. Following GATA's experiment setting (Adhikari et al., 2020), we simplify the game environment by making the action set changeable over time, which can be provided by the TextWorld platform. Note that although the action space is reduced, it still remains challenging as the agent may encounter unseen action candidates (Chandak et al., 2019, 2020). We then use a similar way to obtain a changeable task set, which is a combination of the verb set {chop, dice, slice, fry, make, get, grill, roast} and the ingredient set, where the construction details are provided in Appendix B. Table 4 and Table 5 show the KG-based observations o_t, corresponding subtask candidates $\mathcal{T}$ and action candidates $\mathcal{A}$. Table 6 and Table 7 show more examples of subtasks and actions, respectively. The underlined subtask candidates denote the available subtask set $\mathcal{T}_t$. The underlined action candidates in Table 7 denote the refined action set $\mathcal{A}_t$ after selecting the subtask "roast carrot". We still denote the subtask candidate set (action candidate set) as $\mathcal{T}$ ($\mathcal{A}$) to distinguish it from the available subtask set $\mathcal{T}_t$ (refined action set $\mathcal{A}_t$).

Table 4: The observations o_t, subtask candidates $\mathcal{T}$ and action candidates $\mathcal{A}$ of a simple game and a medium game. The underlined subtask candidates denote the available subtask set $\mathcal{T}_t$.

Game	KG-based observation	Subtask candidates	Action candidates
Simple	["block of cheese", "cookbook", "part_of"], ["block of cheese", "fried", "needs"], ["block of cheese", "player", "in"], ["block of cheese", "raw", "is"], ["block of cheese", "sliced", "needs"], ["block of cheese", "uncut", "is"], ["cookbook", "counter", "on"], ["counter", "kitchen", "at"], ["fridge", "kitchen", "at"], ["fridge", "open", "is"], ["knife", "counter", "on"], ["oven", "kitchen", "at"], ["player", "kitchen", "at"], ["stove", "kitchen", "at"], ["table", "kitchen", "at"]	"fry block of cheese", "get knife", "chop block of cheese", "dice block of cheese", "get block of cheese", "grill block of cheese", "make meal", "roast block of cheese", "slice block of cheese"	"close fridge", "cook block of cheese with oven", "cook block of cheese with stove", "drop block of cheese", "eat block of cheese", "insert block of cheese into fridge", "prepare meal", "put block of cheese on counter", "put block of cheese on stove", "put block of cheese on table", "take cookbook from counter", "take knife from counter"
Medium	["bathroom", "corridor", "south_of"], ["bed", "bedroom", "at"], ["bedroom", "livingroom", "north_of"], ["block of cheese", "cookbook", "part_of"], ["block of cheese", "diced", "is"], ["block of cheese", "diced", "needs"], ["block of cheese", "fridge", "in"], ["block of cheese", "fried", "is"], ["block of cheese", "fried", "needs"], ["carrot", "fridge", "in"], ["carrot", "raw", "is"], ["carrot", "uncut", "is"], ["cookbook", "counter", "on"], ["corridor", "bathroom", "north_of"], ["corridor", "kitchen", "east_of"], ["corridor", "livingroom", "south_of"], ["counter", "kitchen", "at"], ["flour", "cookbook", "part_of"], ["flour", "shelf", "on"], ["fridge", "closed", "is"], ["fridge", "kitchen", "at"], ["frosted-glass door", "closed", "is"], ["frosted-glass door", "kitchen", "west_of"], ["frosted-glass door", "pantry", "east_of"], ["kitchen", "corridor", "west_of"], ["knife", "counter", "on"], ["livingroom", "bedroom", "south_of"], ["livingroom", "corridor", "north_of"], ["oven", "kitchen", "at"], ["parsley", "fridge", "in"], ["parsley", "uncut", "is"], ["player", "kitchen", "at"], ["pork chop", "chopped", "is"], ["pork chop", "chopped", "needs"], ["pork chop", "cookbook", "part_of"], ["pork chop", "fridge", "in"], ["pork chop", "fried", "is"], ["pork chop", "fried", "needs"], ["purple potato", "counter", "on"], ["purple potato", "uncut", "is"], ["red apple", "counter", "on"], ["red apple", "raw", "is"], ["red apple", "uncut", "is"], ["red onion", "fridge", "in"], ["red onion", "raw", "is"], ["red onion", "uncut", "is"], ["red potato", "counter", "on"], ["red potato", "uncut", "is"], ["shelf", "pantry", "at"], ["sofa", "livingroom", "at"], ["stove", "kitchen", "at"], ["table", "kitchen", "at"], ["toilet", "bathroom", "at"], ["white onion", "fridge", "in"], ["white onion", "raw", "is"], ["white onion", "uncut", "is"]	"get block of cheese", "get flour", "get pork chop", "chop block of cheese", "chop flour", "chop pork chop", "dice block of cheese", "dice flour", "dice pork chop", "fry block of cheese", "fry flour", "fry pork chop", "get knife", "grill block of cheese", "grill flour", "grill pork chop", "make meal", "roast block of cheese", "roast flour", "roast pork chop", "slice block of cheese", "slice flour", "slice pork chop"	"go east", "open fridge", "open frosted-glass door", "take cookbook from counter", "take knife from counter", "take purple potato from counter", "take red apple from counter", "take red potato from counter"

Table 5: The observations o_t, subtask candidates $\mathcal{T}$ and action candidates $\mathcal{A}$ of a hard game. The underlined subtask candidates denote the available subtask set $\mathcal{T}_t$. The underlined action candidates denote the refined action set $\mathcal{A}_t$ after selecting the subtask “roast carrot”.

Game	KG-based observation	Subtask candidates	Action candidates
Hard	["backyard", "garden", "west_of"], ["barn door", "backyard", "west_of"], ["barn door", "closed", "is"], ["barn door", "shed", "east_of"], ["bathroom", "corridor", "east_of"], ["bbq", "backyard", "at"], ["bed", "bedroom", "at"], ["bedroom", "corridor", "north_of"], ["bedroom", "livingroom", "south_of"], ["carrot", "cookbook", "part_of"], ["carrot", "player", "in"], ["carrot", "raw", "is"], ["carrot", "roasted", "needs"], ["carrot", "sliced", "needs"],["carrot", "uncut", "is"], ["commercial glass door", "closed", "is"], ["commercial glass door", "street", "east_of"], ["commercial glass door", "supermarket", "west_of"], ["cookbook", "table", "on"], ["corridor", "bathroom", "west_of"], ["corridor", "bedroom", "south_of"], ["counter", "kitchen", "at"], ["driveway", "street", "north_of"], ["fridge", "closed", "is"], ["fridge", "kitchen", "at"], ["front door", "closed", "is"], ["front door", "driveway", "west_of"], ["front door", "livingroom", "east_of"], ["frosted-glass door", "closed", "is"], ["frosted-glass door", "kitchen", "south_of"], ["frosted-glass door", "pantry", "north_of"], ["garden", "backyard", "east_of"], ["kitchen", "livingroom", "west_of"], ["knife", "counter", "on"], ["livingroom", "bedroom", "north_of"], ["livingroom", "kitchen", "east_of"], ["oven", "kitchen", "at"], ["patio chair", "backyard", "at"], ["patio door", "backyard", "north_of"], ["patio door", "corridor", "south_of"], ["patio door", "open", "is"], ["patio table", "backyard", "at"], ["player", "backyard", "at"], ["red apple", "counter", "on"], ["red apple", "raw", "is"], ["red apple", "uncut", "is"], ["red hot pepper", "cookbook", "part_of"], ["red hot pepper", "player", "in"], ["red hot pepper", "raw", "is"], ["red hot pepper", "roasted", "needs"], ["red hot pepper", "sliced", "needs"], ["red hot pepper", "uncut", "is"], ["red onion", "garden", "at"], ["red onion", "raw", "is"], ["red onion", "uncut", "is"], ["shelf", "pantry", "at"], ["showcase", "supermarket", "at"], ["sofa", "livingroom", "at"], ["stove", "kitchen", "at"], ["street", "driveway", "south_of"], ["table", "kitchen", "at"], ["toilet", "bathroom", "at"], ["toolbox", "closed", "is"], ["toolbox", "shed", "at"], ["white onion", "chopped", "needs"], ["white onion", "cookbook", "part_of"], ["white onion", "grilled", "needs"], ["white onion", "player", "in"], ["white onion", "raw", "is"], ["white onion", "uncut", "is"], ["workbench", "shed", "at"], ["yellow bell pepper", "garden", "at"], ["yellow bell pepper", "raw", "is"], ["yellow bell pepper", "uncut", "is"]	"roast carrot", "roast red hot pepper", "grill white onion", "get knife", "chop carrot", "chop red hot pepper", "chop white onion", "dice carrot", "dice red hot pepper", "dice white onion", "fry carrot", "fry red hot pepper", "fry white onion", "get carrot", "get red hot pepper", "get white onion", "grill carrot", "grill red hot pepper", "make meal", "roast white onion", "slice carrot", "slice red hot pepper", "slice white onion"	"go east", "go north", "open barn door", "open patio door", "close patio door", "cook carrot with bbq", "cook red hot pepper with bbq", "cook white onion with bbq", "drop carrot", "drop red hot pepper", "drop white onion", "eat carrot", "eat red hot pepper", "eat white onion", "put carrot on patio chair", "put carrot on patio table", "put red hot pepper on patio chair", "put red hot pepper on patio table", "put white onion on patio chair", "put white onion on patio table"

Table 6: Examples of subtasks.

Subtask candidates		
chop banana	chop black pepper	chop block of cheese
chop olive oil	chop orange bell pepper	chop parsley
chop vegetable oil	chop water	chop white onion
dice cilantro	dice egg	dice flour
dice red bell pepper	dice red hot pepper	dice red onion
dice yellow potato	fry banana	fry black pepper
fry milk	fry olive oil	fry orange bell pepper
fry tomato	fry vegetable oil	fry water
get chicken wing	get cilantro	get egg
get purple potato	get red apple	get red bell pepper
get yellow bell pepper	get yellow onion	get yellow potato
grill green hot pepper	grill lettuce	grill milk
grill salt	grill sugar	grill tomato
roast carrot	roast chicken breast	roast chicken leg
roast peanut oil	roast pork chop	roast purple potato
roast white tuna	roast yellow apple	roast yellow bell pepper
slice green apple	slice green bell pepper	slice green hot pepper
slice red potato	slice red tuna	slice salt

Table 7: Examples of actions.

Action candidates		
chop banana with knife	chop block of cheese with knife	chop carrot with knife
cook block of cheese with oven	cook block of cheese with stove	cook carrot with bbq
cook orange bell pepper with oven	cook orange bell pepper with stove	cook parsley with bbq
cook water with stove	cook white onion with bbq	cook white onion with oven
drink water	drop banana	drop black pepper
eat carrot	eat chicken breast	eat chicken leg
insert block of cheese into toolbox	insert carrot into fridge	insert carrot into toolbox
insert red onion into fridge	insert red onion into toolbox	insert red potato into fridge
put banana on shelf	put banana on showcase	put banana on sofa
put chicken breast on showcase	put chicken breast on sofa	put chicken breast on stove
put egg on patio table	put egg on shelf	put egg on showcase
put green hot pepper on shelf	put green hot pepper on showcase	put green hot pepper on sofa
put olive oil on patio chair	put olive oil on patio table	put olive oil on shelf
put pork chop on sofa	put pork chop on stove	put pork chop on table
put red hot pepper on table	put red hot pepper on toilet	put red hot pepper on workbench
put salt on workbench	put sugar on bed	put sugar on counter
put white onion on shelf	put white onion on showcase	put white onion on sofa
put yellow onion on sofa	put yellow onion on stove	put yellow onion on table
take banana from patio chair	take banana from patio table	take banana from shelf
take carrot from showcase	take carrot from sofa	take carrot from stove
take chicken wing from toolbox	take chicken wing from workbench	take cilantro
take green apple from bed	take green apple from counter	take green apple from fridge
take lettuce from sofa	take lettuce from stove	take lettuce from table
take orange bell pepper from workbench	take parsley	take parsley from bed
take purple potato from showcase	take purple potato from sofa	take purple potato from stove
take red hot pepper from toolbox	take red hot pepper from workbench	take red onion
take salt from counter	take salt from fridge	take salt from patio chair
take water from counter	take water from fridge	take water from patio chair
take yellow apple from sofa	take yellow apple from stove	take yellow apple from table

B Pre-training Datasets

We build separate datasets for each pre-training task (task decomposition, action pruning, and imitation learning). We first let the player to go through each simple game, then construct the datasets upon the interaction data. For each time step, the game environment provides the player with the action set $\mathcal{A}$ and the KG-based observation o_t, which is represented as a set of triplets. We use a simple method to build the subtask set $\mathcal{T}$ from o_t: As shown in Fig. 8, we first obtain the ingredients by extracting the nodes having the relation "part_of" with the node "cookbook". Then we build $\mathcal{T}$ as the Cartesian product of the ingredients and the verbs {chop, dice, slice, fry, get, grill, roast} plus two special subtasks "get knife" and "make meal". The player is required to select a subtask $T_t \in \mathcal{T}$, and select an action $a_t \in \mathcal{A}$. After executing a_t, the environment will transit to next state s_{t+1}, and the player will receive o_{t+1} and r_{t+1} to form a transition $\{o_t, \mathcal{T}, T_t, \mathcal{A}, a_t, o_{t+1}, r_{t+1}\}$, where $\{o_t, \mathcal{T}, T_t, \mathcal{A}, a_t\}$ will be used for imitation learning. Fig. 8 shows the construction process of the pre-training dataset for task decomposition. Each subtask candidate $T \in \mathcal{T}$ will formulate a question "Is T available?", whose answer is 1 (yes) if T is an available subtask for o_t, otherwise 0 (no). Fig. 9 shows the construction process of the pre-training dataset for action pruning. The action selector is made invariant of o_t, that we consider every subtask candidate $T \in \mathcal{T}$ during pre-training, regardless of whether T is a currently-available subtask. Each action candidate $a \in \mathcal{A}$ will be paired with T to formulate a question "Is a relevant to T", whose answer is 1 if a is relevant to T, otherwise 0.

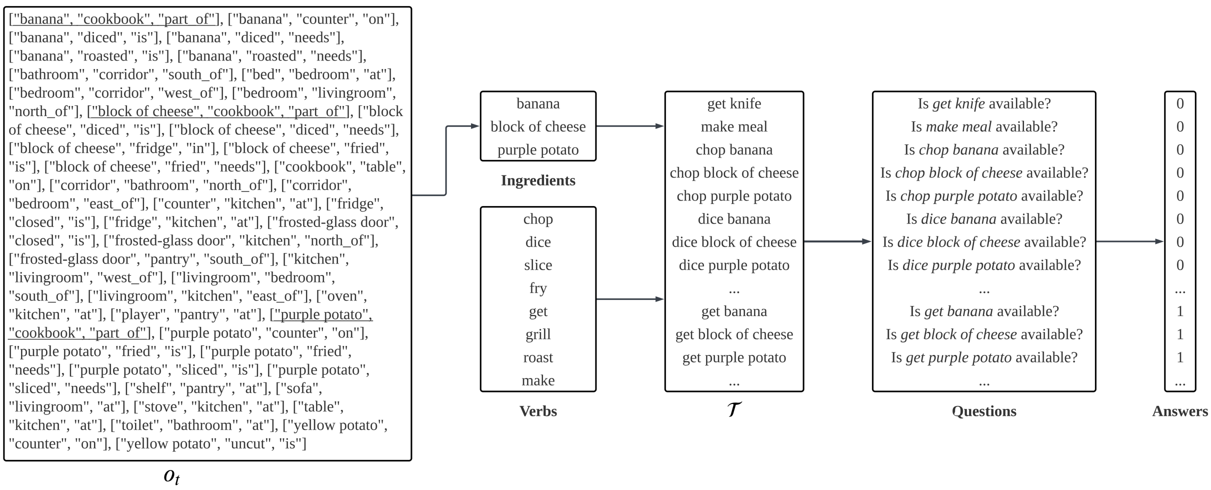

Figure 8: The construction process of the subtask set $\mathcal{T}$, and the pre-training dataset for task decomposition.

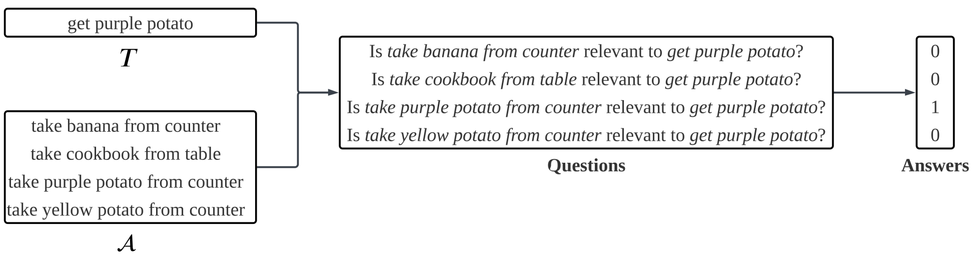

Figure 9: The construction process of the pre-training dataset for action pruning.

C Baseline details

C.1 GATA

Fig. 10 shows our backbone model GATA, which consists of an observation encoder, an action encoder and an action scorer. The observation encoder is a graph encoder for encoding the KG-based observation o_t, and the action encoder is a text encoder to encode the action set $\mathcal{A}$ as a stack of action candidate representations. The observation representation will be paired with each action candidate, and then fed into the action scorer, which consists of linear layers.

We train the GATA through reinforcement learning, the experiment setting is same with Sec. 5.3. Instead of initializing the word embedding, node embedding and edge embedding with fastText word vectors (Mikolov et al., 2017), we found that the action prediction task (AP), which is also included in GATA's work (Adhikari et al., 2020), could provide better initialization. In light of this, we could like to conduct such task, and apply the AP initialization to all encoders (observation encoder, task encoder, action encoder). Fig. 11 shows the action predicting process. Given the transition data, the task is to predict the action $a_t \in \mathcal{A}$ given the current observation o_t, and the next observation o_{t+1} after executing a_t. The transition data for AP task is collected from the FTWP game set and is provided by GATA's released code.

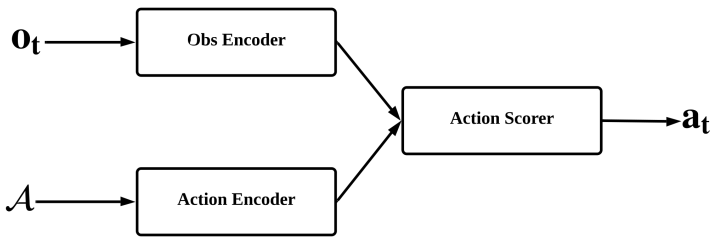

Figure 10: The architecture of GATA baseline.

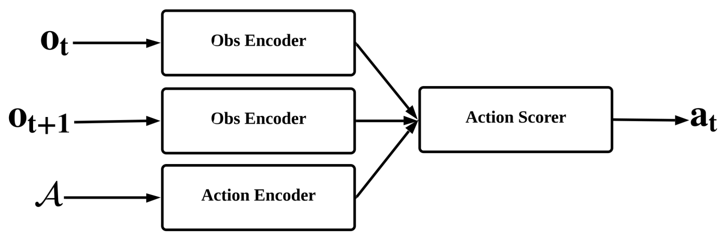

Figure 11: The architecture of GATA for action prediction.

C.2 IL

Fig. 12 shows the IL baseline. We follow (Chen et al., 2021) to conduct a two-phase training process: imitation pre-training and reinforcement fine-tuning. In the imitation pre-training phase, we use the transition data to train both the task selector ($f(o_t, \mathcal{T}) \to T_t$) and the action selector ($f(o_t, T_t, \mathcal{A}) \to a_t$) through supervised learning. The modules are optimized via cross entropy loss and Adam optimizer with learning rate 0.001. We train the modules with batch size 128 for up to 50 epochs. Then in the reinforcement fine-tuning phase, we freeze the task selector and fine-tune the action selector through reinforcement learning, where the experiment setting is same with QWA and GATA.

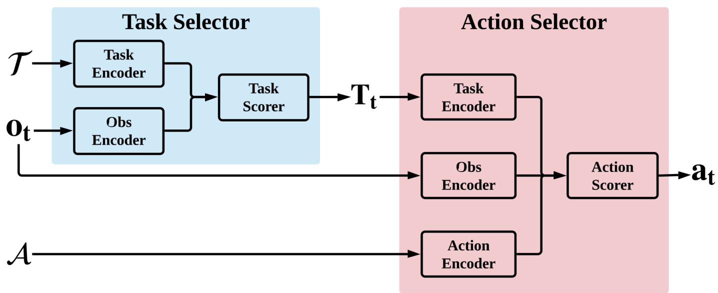

Figure 12: The architecture of IL baseline.

D More experimental results

In the pre-training phase, we conduct rough hyper-parameter tuning by varying batch sizes. Fig. 13 and Fig. 14 show the pre-training performance of QWA's task selector and action validator, respectively. Fig. 15 shows the pre-training performance of IL baseline.

Fig. 16 compares our GATA and the original GATA without the action prediction initialization. Fig. 17, Fig. 18, Fig. 19 and Fig. 20 show the full results of Fig. 4, Fig. 5, Fig. 6 and Fig. 7, respectively.

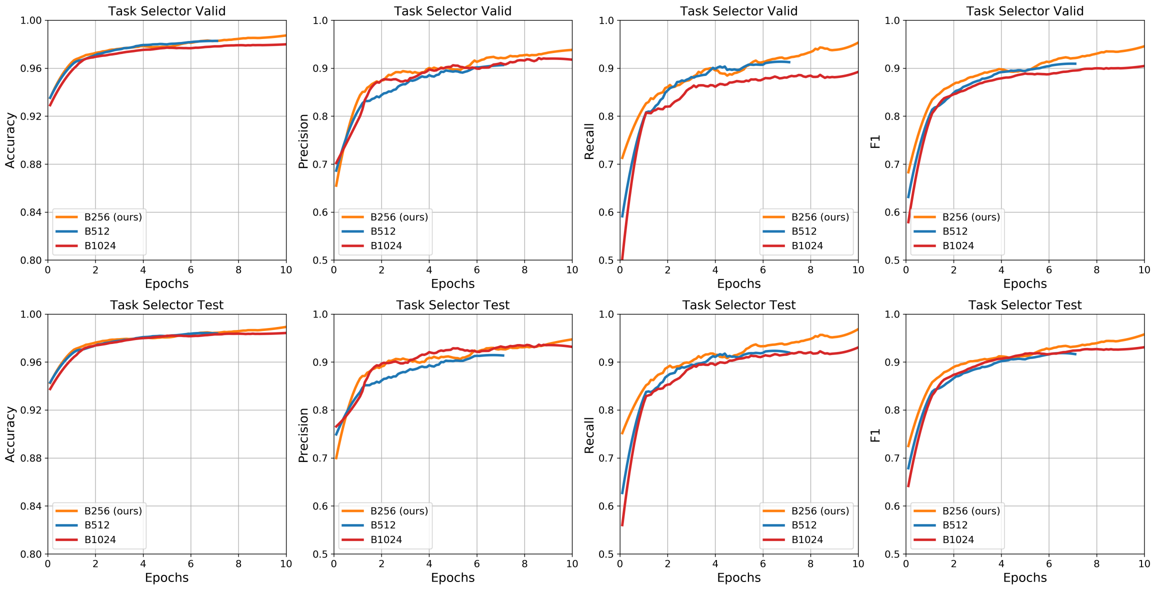

Figure 13: The pre-training performance of QWA's task selector. The results are averaged by 3 random seeds, we omit the standard deviation as the performance is relatively stable.

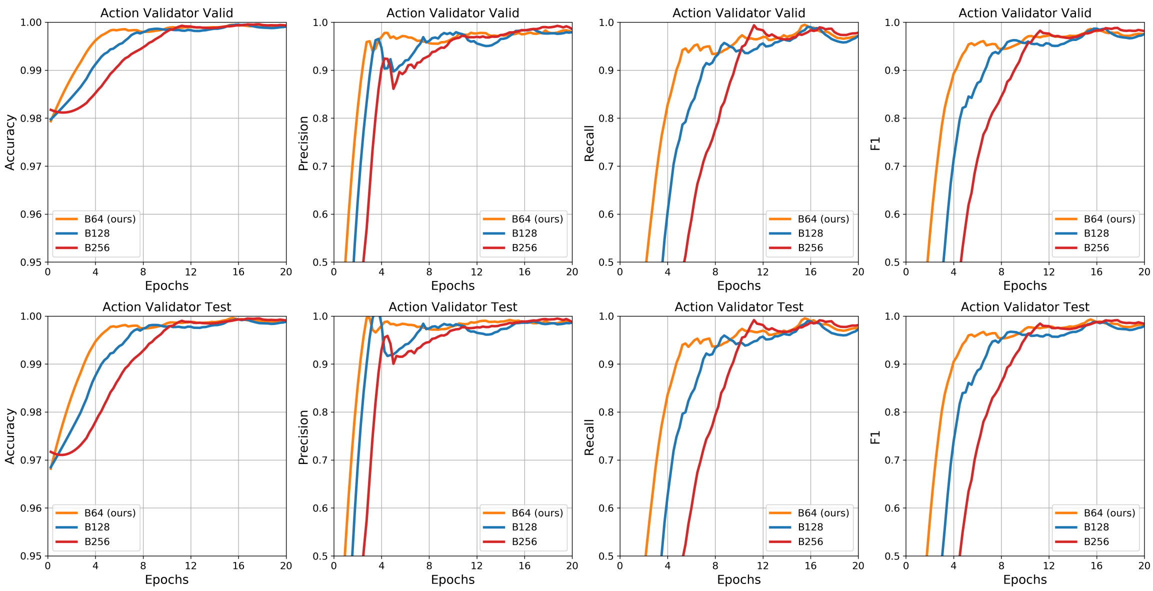

Figure 14: The pre-training performance of QWA's action validator.

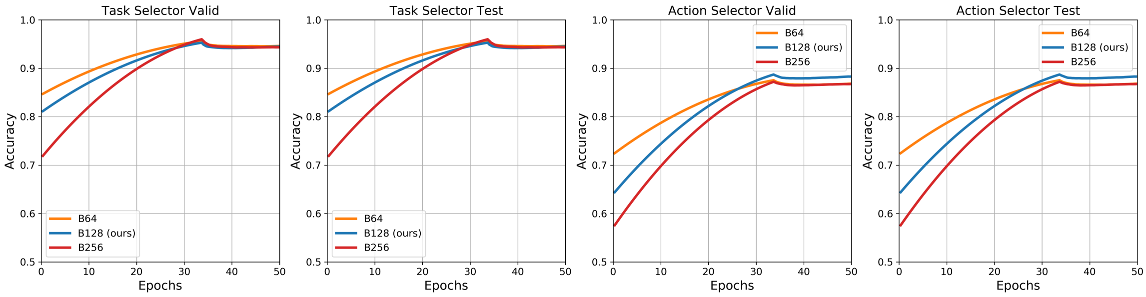

Figure 15: The pre-training performance of IL's task selector and action selector.

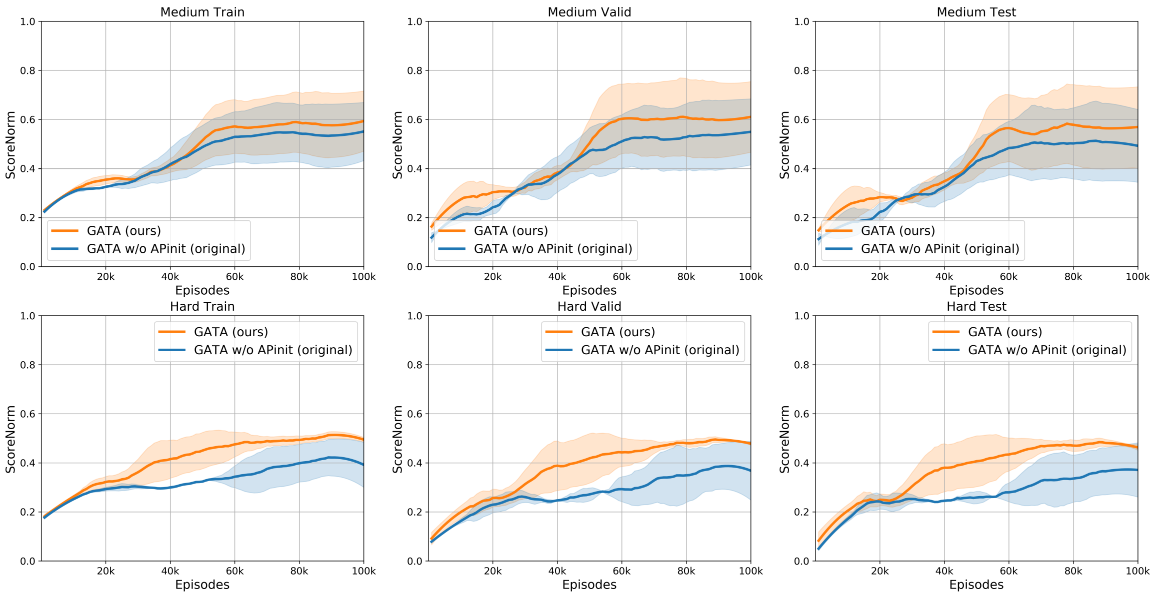

Figure 16: The RL performance of our GATA baseline and the original GATA without AP initialization.

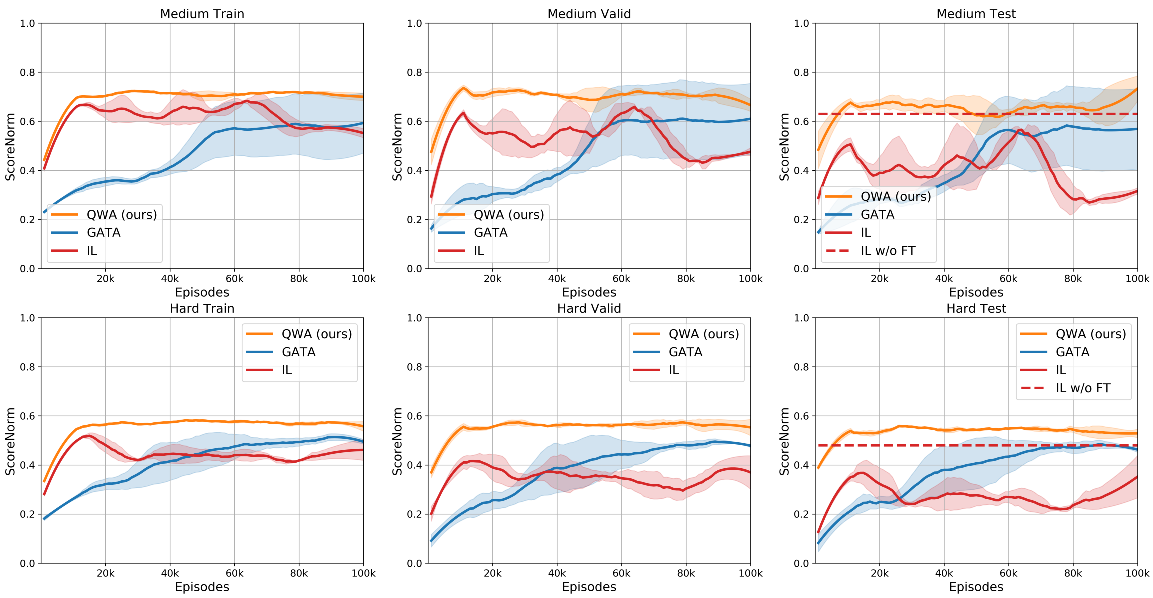

Figure 17: The RL performance of models with respect to training episodes (the full result of Fig. 4).

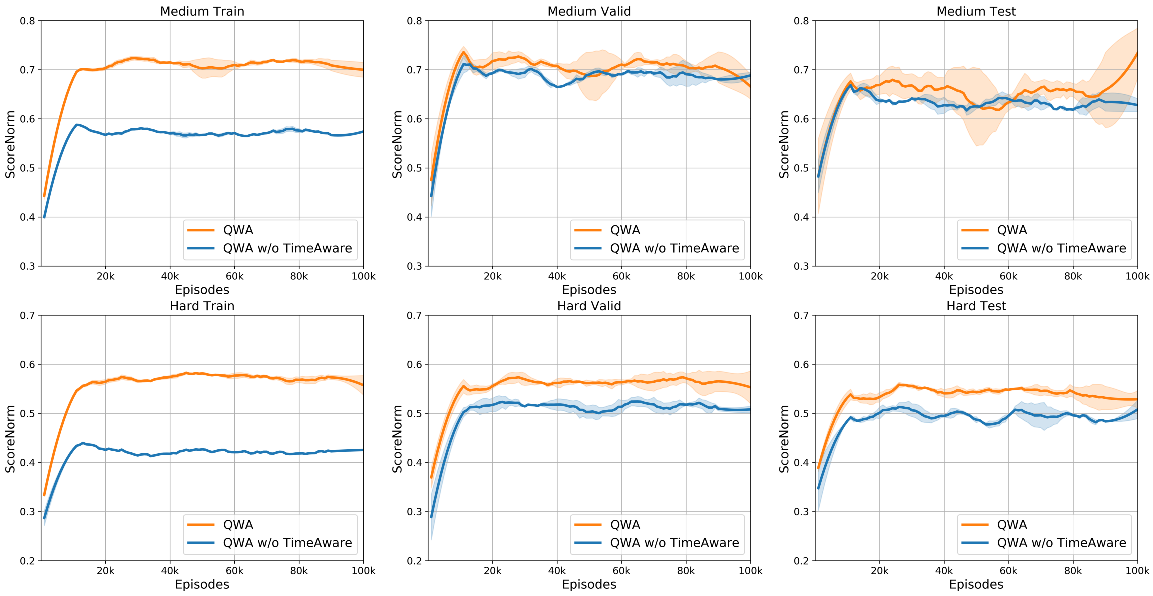

Figure 18: The RL performance of our model and the variant without time-awareness (the full result of Fig. 5).

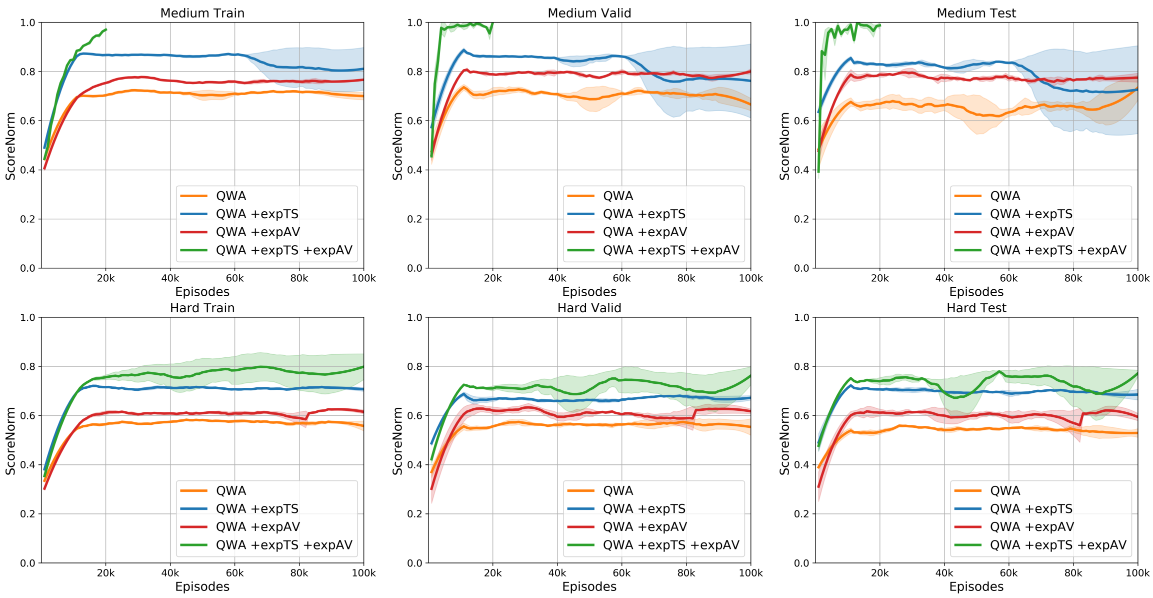

Figure 19: The performance of our model and the variants with expert modules (the full result of Fig. 6).

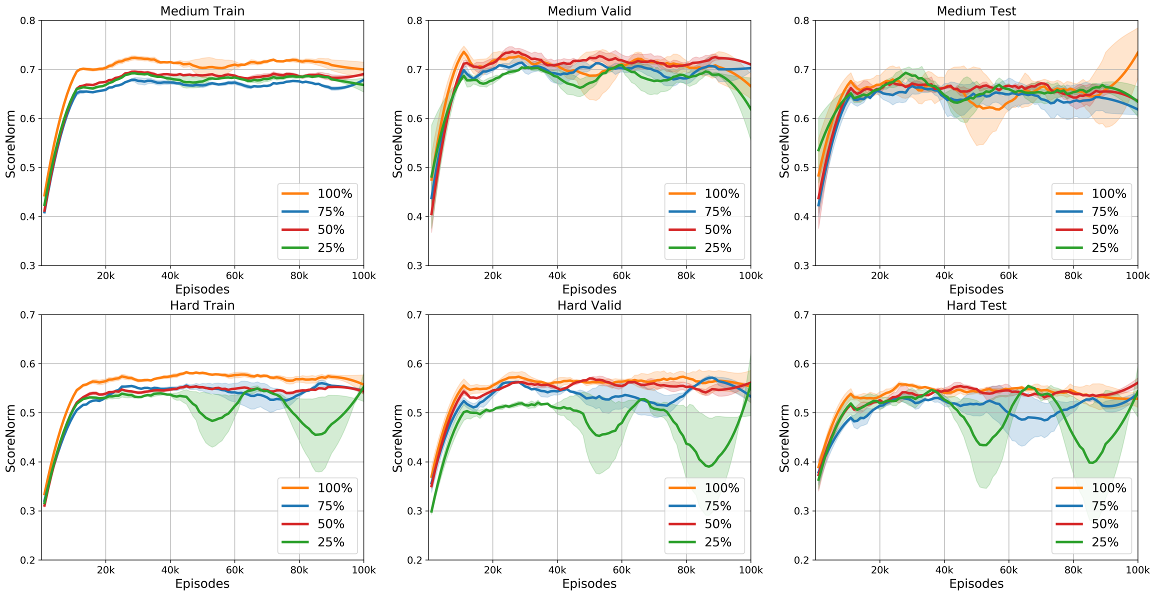

Figure 20: The performance of our model with varying amounts of pre-training data (the full result of Fig. 7).